$X \in \mathbb{R}^{5000 \times 400}$.

$y \in \mathbb{R}^{5000 \times 1}$.

```
function [J grad] = nnCostFunction(nn_params, ...
                                   input_layer_size, ...
                                   hidden_layer_size, ...
                                   num_labels, ...
                                   X, y, lambda)
%NNCOSTFUNCTION Implements the neural network cost function for a two layer
%neural network which performs classification
%   [J grad] = NNCOSTFUNCTON(nn_params, hidden_layer_size, num_labels, ...
%   X, y, lambda) computes the cost and gradient of the neural network. The
%   parameters for the neural network are "unrolled" into the vector
%   nn_params and need to be converted back into the weight matrices.
%
%   The returned parameter grad should be a "unrolled" vector of the
%   partial derivatives of the neural network.
%

% Reshape nn_params back into the parameters Theta1 and Theta2, the weight matrices
% for our 2 layer neural network
Theta1 = reshape(nn_params(1:hidden_layer_size * (input_layer_size + 1)), ...
                 hidden_layer_size, (input_layer_size + 1));

Theta2 = reshape(nn_params((1 + (hidden_layer_size * (input_layer_size + 1))):end),
...
                 num_labels, (hidden_layer_size + 1));

% Setup some useful variables
m = size(X, 1);
```

number of training examples.

```
% You need to return the following variables correctly
J = 0;
Theta1_grad = zeros(size(Theta1));
Theta2_grad = zeros(size(Theta2));
```

X = [ones(m, 1)   X]

```
% ====================== YOUR CODE HERE ======================
% Instructions: You should complete the code by working through the
%               following parts.
%
% Part 1: Feedforward the neural network and return the cost in the
%         variable J. After implementing Part 1, you can verify that your
%         cost function computation is correct by verifying the cost
%         computed in ex4.m
%
% Part 2: Implement the backpropagation algorithm to compute the gradients
%         Theta1_grad and Theta2_grad. You should return the partial derivatives of
%         the cost function with respect to Theta1 and Theta2 in Theta1_grad and
%         Theta2_grad, respectively. After implementing Part 2, you can check
%         that your implementation is correct by running checkNNGradients
%
%         Note: The vector y passed into the function is a vector of labels
%               containing values from 1..K. You need to map this vector into a
%               binary vector of 1's and 0's to be used with the neural network
%               cost function.
%
%         Hint: We recommend implementing backpropagation using a for-loop
%               over the training examples if you are implementing it for the
%               first time.
%
% Part 3: Implement regularization with the cost function and gradients.
%
```

```
%         Hint: You can implement this around the code for
%               backpropagation. That is, you can compute the gradients for
%               the regularization separately and then add them to Theta1_grad
%               and Theta2_grad from Part 2.
%
```

% Part (I) $J(\Theta) = \frac{-1}{m} \sum_{i=1}^{m} \sum_{k=1}^{K} [+ y_k^{(i)} \log(\underbrace{h_\Theta(x^{(i)})}_{= g(\Theta^T x)}]_k) + (1 - y_k^{(i)}) \cdot \log(1 - h_\Theta(x^{(i)})_k)]$.

for i=1:m % loop through examples.

new_y = recode_y (y(m), k);

J+= sum (y.* log (sigmoid (Theta1 * X)

row.

$y = \begin{pmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \in \mathbb{R}^{10}$.

for single example

$\underline{x} = \begin{pmatrix} | \\ x_1 \\ | \end{pmatrix}$, $y = \begin{pmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$

```
% ---------------------------------------------------------------

% =========================================================================

% Unroll gradients
grad = [Theta1_grad(:) ; Theta2_grad(:)];

end
```

$x \begin{pmatrix} | \\ | \\ x \\ | \end{pmatrix} \rightarrow h \in \mathbb{R}^K$.

$\begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \cdot \log(h) + (1 - \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix})$ $\log(1-h)$

sum over col.

function new_y = recode_y (y, k).

new_y = ones (k, 1)

new_y (k) = 1

end

$\Rightarrow$ $X = \begin{pmatrix} — x_1 — \\ — x_2 — \\ \vdots \\ — x_m — \end{pmatrix}$, $y = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 1 & 0 \\ & & \vdots & & \end{pmatrix}$

$h = \begin{pmatrix} — h_1 — \\ \vdots \\ — h_m \end{pmatrix}$

$\begin{pmatrix} 1 & — x_1 — \\ 1 & — x_2 — \\ \vdots & \\ 1 & — x_m — \end{pmatrix} \begin{pmatrix} | & | & & | \\ \theta_{11} & \theta_{12} & \cdots & \theta_{1..} \\ | & | & & | \end{pmatrix} \rightarrow \begin{pmatrix} a_1^{(2)} & a_2^{(2)} & \cdots & a^{(2)} \end{pmatrix} \rightarrow$ sample 1

add 1

$\Rightarrow$ $y .* h$ sum over row, then col.